Spartan-6 LX9 MicroBoard Embedded Tutorial

Tutorial 4

Embedded System Simulation

Version 13.1.01

Revision History

Version	Description	Date
13.1.01	Initial release for EDK 13.1	5/17/2011

Table of Contents

Table of Figures

Overview

This is the fourth tutorial in a series of training material dedicated to introducing engineers to creating their first embedded designs. These tutorials will cover all the required steps for creating a complete MicroBlaze design in the Spartan-6 LX9 MicroBoard. While dedicated to this platform, the information learned here can be used with any Xilinx FPGA.

The tutorial is divided into three main steps: Setting up the simulation environment, adding a testbench file, and using ISim to simulate the system. The test application will reside in Block memory to provide a cycle accurate simulation. We will be using the design from the last tutorial.

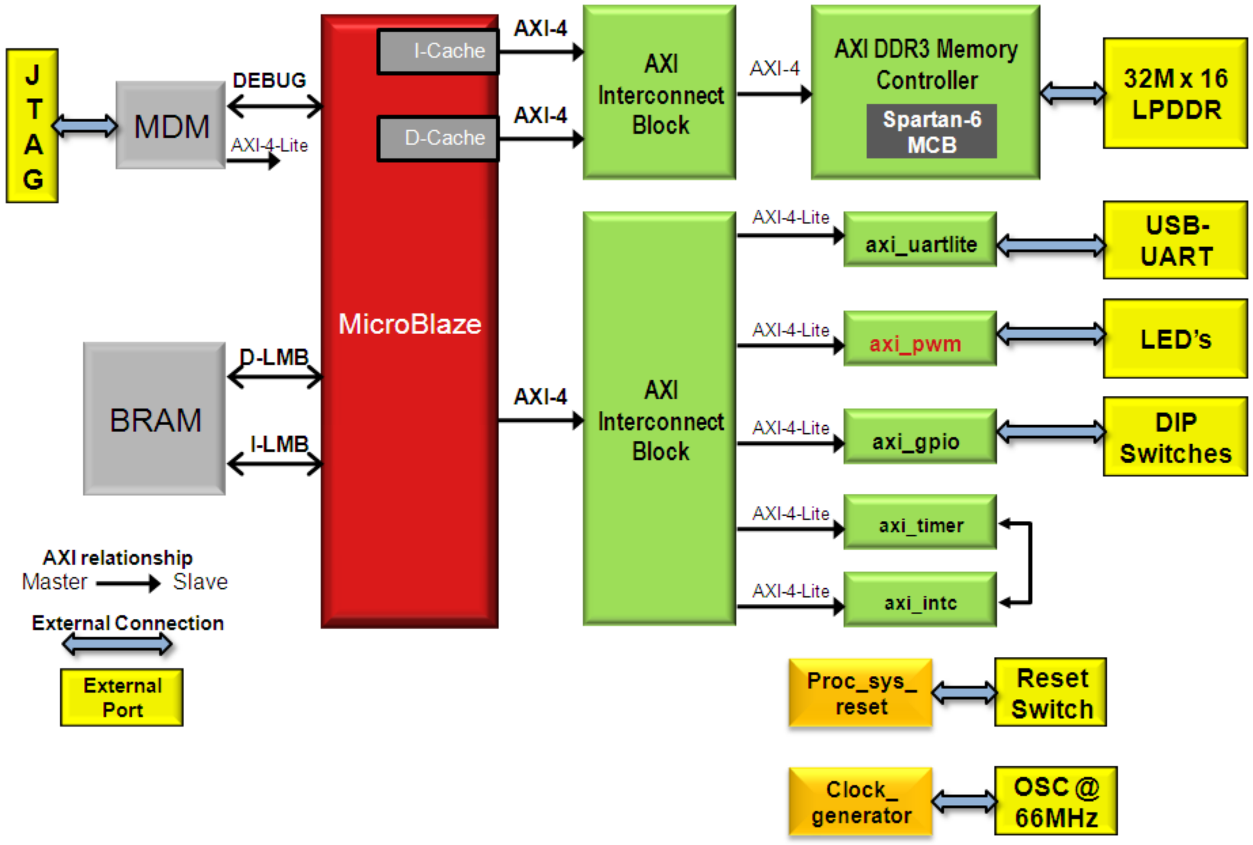


Figure 1 - Hardware Platform

Objectives

This tutorial demonstrates how to simulate the embedded system using ISim. The tutorial will show

- How to setup the simulation properties
- How to add a VHDL testbench
- How to simulate a cycle accurate MicroBlaze design
- How to view MicroBlaze specific signals

Requirements

The following items are required for proper completion of this tutorial.

- Completion of the Adding Custom IP to an Embedded System Tutorial.

Software

The following software setup is required to test this reference design:

- WindowsXP 32-bit Service Pack 2
- Xilinx ISE WebPack with the EDK add-on or ISE Embedded Edition version 13.1
- Installed Digilent Adept and Xilinx 3rd-party USB Cable driver (see *Spartan-6 LX9 MicroBoard Configuration Guide*, listed in Recommended Reading, below)
- Installed Silicon Labs CP210x USB-to-UART Bridge Driver (see *Silicon Labs CP210x USB-to-UART Setup Guide*, listed in Recommended Reading, below)
- Installation of the Spartan-6 LX9 MicroBoard IPXACT files (Available from Avnet: *http://em.avnet.com/s6microboard*)

Hardware

The hardware setup used by this reference design includes:

- Computer with a minimum of 300-900 MB (depending on O/S) to complete an XC6SLX9 design[1]
- Avnet Spartan-6 LX9 MicroBoard Kit
  - Avnet Spartan-6 LX9 MicroBoard
  - USB Extension cable (if necessary)
  - USB A-to-MicroB cable

Recommended Reading

Available from Avnet: *http://em.avnet.com/s6microboard*

- The hardware used on the Spartan-6 LX9 MicroBoard is described in detail in Avnet document, *Spartan-6 LX9 MicroBoard User Guide*.
- An overview of the configuration options available on the Spartan-6 LX9 MicroBoard, as well as Digilent driver installation instructions can be found in the Avnet document, *Spartan-6 LX9 MicroBoard Configuration Guide.*
- Instructions on installing the Silicon Labs CP210x USB-to-UART drivers can be found in the Avnet document, *Silicon Labs CP210x USB-to-UART Setup Guide*.

Available from Xilinx: *http://www.xilinx.com/support/documentation/spartan-6.htm*

- Details on the Spartan-6 FPGA family are included in the following Xilinx documents:
  - *Spartan-6 Family Overview (DS160)*
  - *Spartan-6 FPGA Data Sheet (DS162)*
  - *Spartan-6 FPGA Configuration User Guide (UG380)*
  - *Platform Studio Help (available in tool menu)*
  - *Platform Studio SDK Help (available in tool menu)*
  - *MicroBlaze Reference Guide v.13.1 (UG081)*
  - *Embedded System Tools Reference Manual v.13.1 (UG111)*

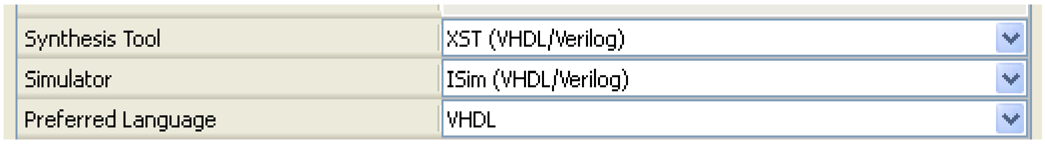
[1] Refer to www.xilinx.com/ise/products/memory.htm

I. Setting up the Simulation Environment

Very little setup is required to simulate an embedded design from ISE. We just need to verify that ISim is selected as the main hardware simulator and add the Tutorial_Test ELF file created in SDK.

1) Start ISE Project Navigator and open the **EDK_Tutorial** project.

2) Go to **Project > Design Properties** and verify that ISim is selected for simulation. Click **OK**.

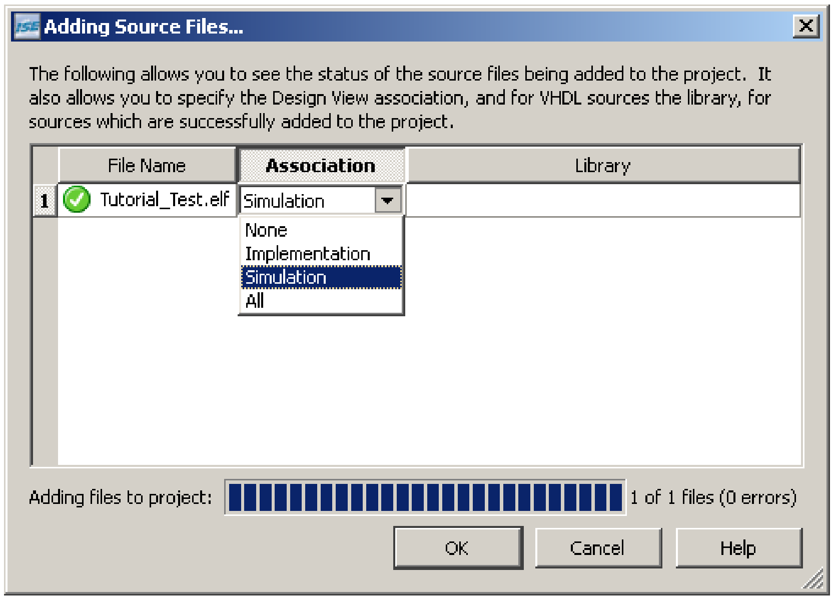


Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL

Figure 2 - Selecting ISim in Design Properties

3) Go to **Project > Add Source.** Browse to the \EDK_Tutorial\WorkSpace\Tutorial_Test\Debug\ directory and open **Tutorial_Test.elf**.

4) Set the associations to **Simulation** and click **OK**.



Adding Source Files...

The following allows you to see the status of the source files being added to the project. It also allows you to specify the Design View association, and for VHDL sources the library, for sources which are successfully added to the project.

	File Name	Association	Library
1	Tutorial_Test.elf	Simulation	

Adding files to project: 1 of 1 files (0 errors)

Figure 3 - Adding an ELF file for Simulation

5) Check the box **Use With** box to associate the ELF file to the MicroBlaze processor for simulation. Click **OK**.

II. Adding a Test Bench File

To simulate our design we will need to add a VHDL test bench. The testbench will instantiate our top level module and provide stimulis for the input ports.

1) In Project Navigator, go to **Project > New Source.**
2) Select **VHDL Test Bench** and select **Testbench** for the file name. Click **Next**.
3) Select **mb_system_top** and click **Next**. Click **Finish**.
4) Click on the **Simulation View** radio button. The type of simulation can be changed by using the drop-down list. We will do a Behavioral simulation.

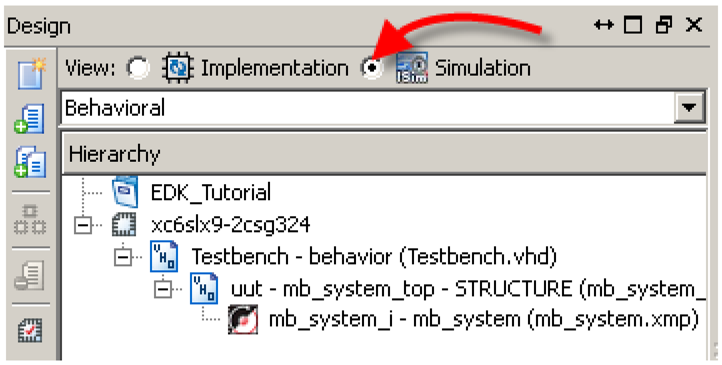
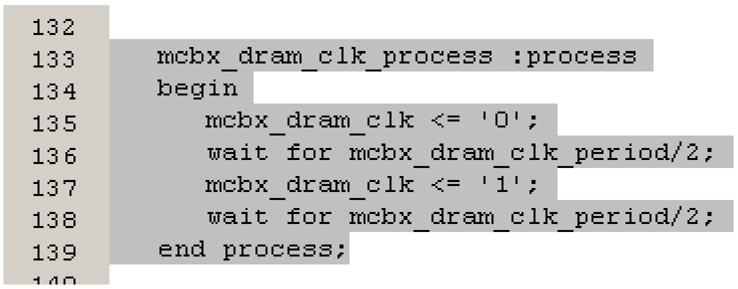

Design
View:
Implementation
Simulation
Behavioral
Hierarchy
EDK_Tutorial
xc6slx9-2csg324
Testbench - behavior (Testbench.vhd)
uut - mb_system_top - STRUCTURE (mb_system_
mb_system_i - mb_system (mb_system.xmp)

Figure 4 - Selecting Simulation View

5) The test bench should be open in the Editor.
6) Delete the **mcbx_dram_clk_process** from the process line to the end process line.



```
   mcbx_dram_clk_process :process
   begin
      mcbx_dram_clk <= '0';
      wait for mcbx_dram_clk_period/2;
      mcbx_dram_clk <= '1';
      wait for mcbx_dram_clk_period/2;
   end process;
```

Figure 5 - Remove LPDDR Memory Clock Process

7) Change the constant for **CLK_66MHZ_period** from 10 ns to 15 ns to reflect our 66 MHz input clock period.
8) Add stimulus for the reset, and DIP switches. Reset_in is active high on the MicroBoard.

   After - - hold reset state for 100 ns, add:

```
RESET <= '1';
```

After wait for 100 ns, add:

```
RESET <= '0';
DIP_Switches_GPIO_IO_I_pin <= "1000";
```

After -- insert stimulus here, add:

```
wait for 70 us;
DIP_Switches_GPIO_IO_I_pin <= "0001";
```

9) Save the test bench file.

10) Click on **Testbench – behavior (Testbench.vhd)** in the Hierarchy Window. Then in the Processes expand **ISim Simulator** and double-click **Behavioral Check Syntax**. This will check to see if you have any errors in your testbench.

III. Simulating the System Using ISim

The simulation can take advantage of code running from BRAMs, providing a cycle accurate MicroBlaze simulation. We will be able to see and trace MicroBlaze execution. ISim will use the software application selected to initialize in BRAMs in XPS. We will select the Tutorial_Test (ELF) application from the last tutorial in SDK.

We need to modify the application in SDK since writing to the UART would take too long in a simulation environment. We will need to comment out the print statement. Additionally, we need to mark the application, Tutorial_Test, to initialize to BRAMs.

1) Start Xilinx **SDK** and select the Workspace from **EDK_Tutorial**.

2) Open the **Tutorial_Test main.c** source file and comment out the print statement by adding **//** at the beginning of the line.

3) Save the main.c file. This will automatically compile the new ELF file.

4) Close SDK.

5) In Project Navigator, select **Testbench** in the hierarchy view.

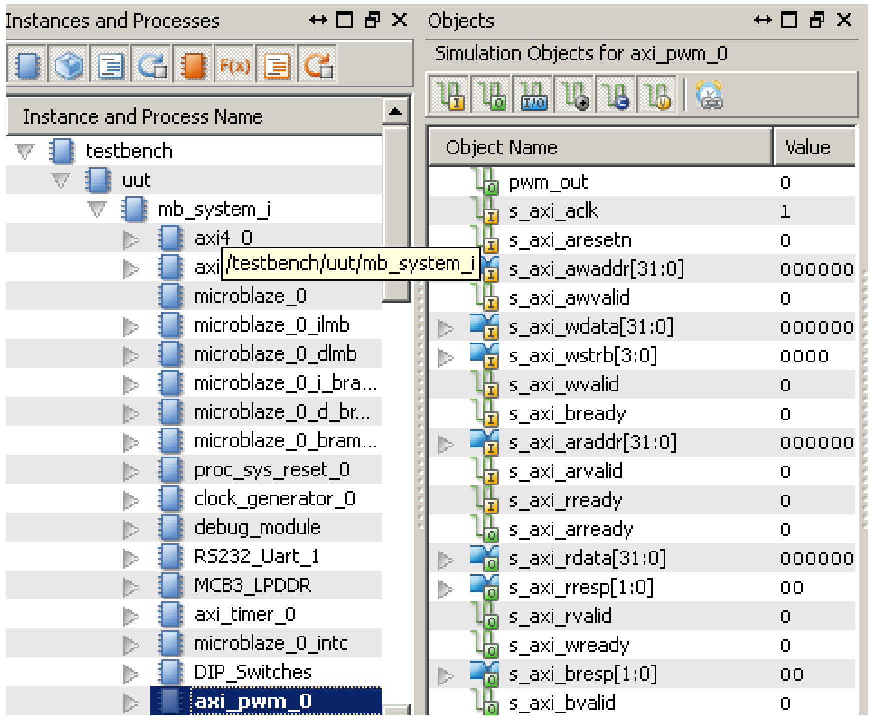
6) In the **Processes** window, double-click on **Simulate Behavioral Model**. The tools will load and compile all the necessary files.

7) Wait for ISim to open.

8) We will add some internal signals to the waveform. Select the **Instances and Processes** tab on the left side.

9) Expand **testbench** then **UUT** then **mb_system_i** to view the embedded system components.

10) Select **axi_pwm_0** and drag it over to the waveform window. In the Objects window, you can also expand the **axi_pwm_0** instance and drill down a couple levels to view the **user_logic** signals. **FCOUNT** and **DUTY_CYCLE** are valuable to see in the waveform window.



Instance and Process Name
- testbench
  - uut
    - mb_system_i
      - axi4_0
      - axi /testbench/uut/mb_system_i
      - microblaze_0
      - microblaze_0_ilmb
      - microblaze_0_dlmb
      - microblaze_0_i_bra...
      - microblaze_0_d_br...
      - microblaze_0_bram...
      - proc_sys_reset_0
      - clock_generator_0
      - debug_module
      - RS232_Uart_1
      - MCB3_LPDDR
      - axi_timer_0
      - microblaze_0_intc
      - DIP_Switches
      - axi_pwm_0

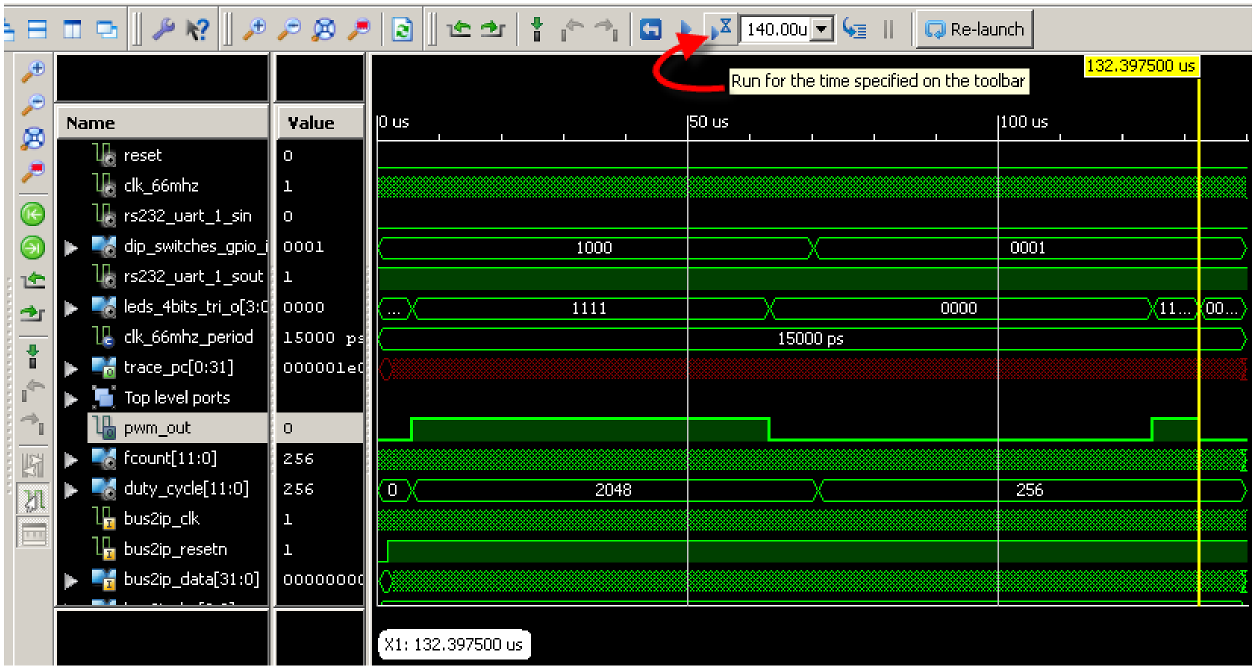
Object Name	Value
pwm_out	0
s_axi_aclk	1
s_axi_aresetn	0
s_axi_awaddr[31:0]	000000
s_axi_awvalid	0
s_axi_wdata[31:0]	000000
s_axi_wstrb[3:0]	0000
s_axi_wvalid	0
s_axi_bready	0
s_axi_araddr[31:0]	000000
s_axi_arvalid	0
s_axi_rready	0
s_axi_arready	0
s_axi_rdata[31:0]	000000
s_axi_rresp[1:0]	00
s_axi_rvalid	0
s_axi_wready	0
s_axi_bresp[1:0]	00
s_axi_bvalid	0

Figure 6 - Adding Simulation Nets from UUT

11) Select microblaze_0 to view all the MicroBlaze objects. There are a lot valuable signals which can be observed during simulation. An example would be the program counter. In the Object window, scroll-down to the **trace_pc[0:31]** signal. Select the signal and drag it over to the waveform window.

12) To simplify the simulation waveform window, you can delete any of the mcbx signals since we are not utilizing the LPDDR in this tutorial.

13) Run the simulation for 140 us. In the console window, type: run 140us. This may take a few minutes to load.



Name	Value
reset	0
clk_66mhz	1
rs232_uart_1_sin	0
dip_switches_gpio_i	0001
rs232_uart_1_sout	1
leds_4bits_tri_o[3:0	0000
clk_66mhz_period	15000 ps
trace_pc[0:31]	000001e0
Top level ports	
pwm_out	0
fcount[11:0]	256
duty_cycle[11:0]	256
bus2ip_clk	1
bus2ip_resetn	1
bus2ip_data[31:0]	00000000

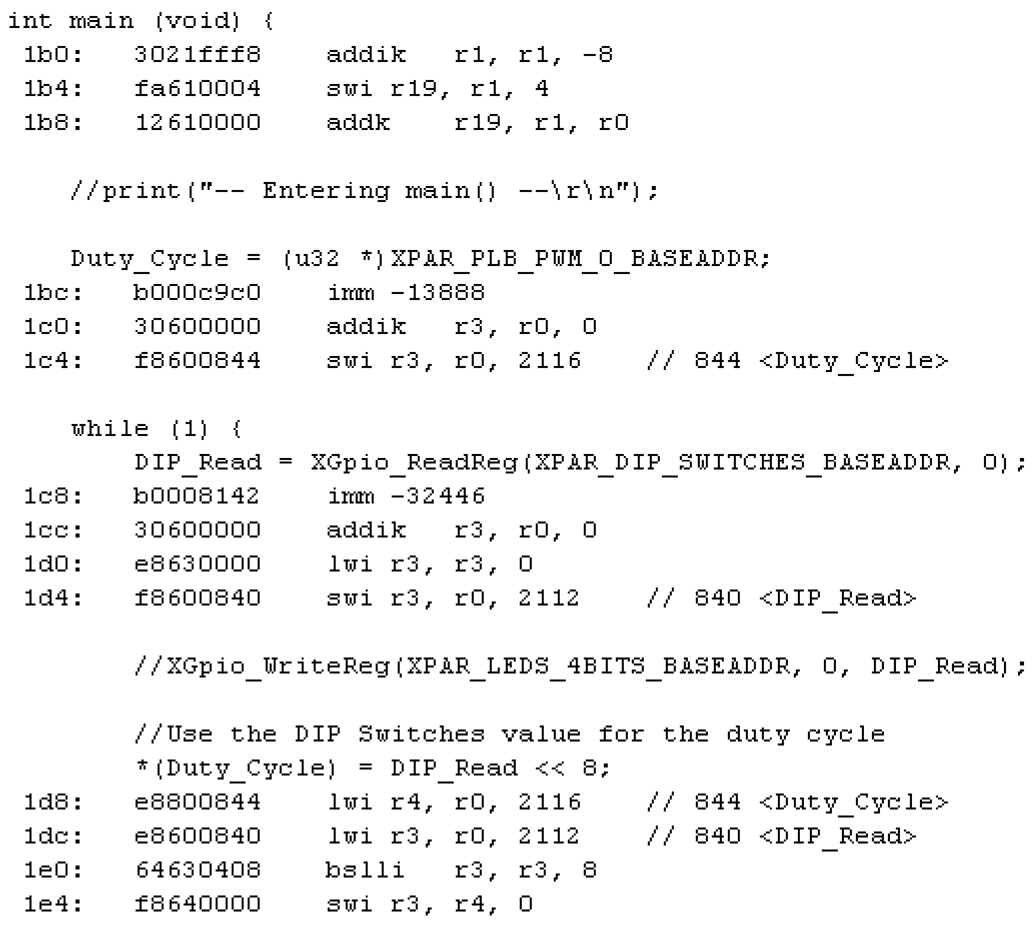
Figure 7 - Run Simulation for 80us

14) Look at the cycles on the AXI bus and observe the **pwm_out** signal. You can see it goes valid twice during this period. In the simulation testbench, we've set to duty cycle widths and they are shown here. For the first PWM cycle, the duty cycle is set to 2048, and **pwm_out** is valid until **fcount** exceeds that value. In the second PWM pulse, the duty cycle is set to 256.

15) You can also correlate the program counter (trace_pc) to the C application.

16) In SDK, expand the **Tutorial_Test/Debug** project. Double-click on the **Tutorial_Test.elf** executable file.

17) Scroll down to view the disassembly of the C code.



```
int main (void) {
 1b0:   3021fff8    addik   r1, r1, -8
 1b4:   fa610004    swi r19, r1, 4
 1b8:   12610000    addk    r19, r1, r0

    //print("-- Entering main() --\r\n");

    Duty_Cycle = (u32 *)XPAR_PLB_PWM_0_BASEADDR;
 1bc:   b000c9c0    imm -13888
 1c0:   30600000    addik   r3, r0, 0
 1c4:   f8600844    swi r3, r0, 2116    // 844 <Duty_Cycle>

    while (1) {
        DIP_Read = XGpio_ReadReg(XPAR_DIP_SWITCHES_BASEADDR, 0);
 1c8:   b0008142    imm -32446
 1cc:   30600000    addik   r3, r0, 0
 1d0:   e8630000    lwi r3, r3, 0
 1d4:   f8600840    swi r3, r0, 2112    // 840 <DIP_Read>

        //XGpio_WriteReg(XPAR_LEDS_4BITS_BASEADDR, 0, DIP_Read);

        //Use the DIP Switches value for the duty cycle
        *(Duty_Cycle) = DIP_Read << 8;
 1d8:   e8800844    lwi r4, r0, 2116    // 844 <Duty_Cycle>
 1dc:   e8600840    lwi r3, r0, 2112    // 840 <DIP_Read>
 1e0:   64630408    bslli   r3, r3, 8
 1e4:   f8640000    swi r3, r4, 0
```

Figure 8 - Disassembly of C Code

18) In the ISim simulation window, look at the **trace_pc** bus (change the radius to HEX). The PC goes between 0x1C8 and 0x1E4 which are the instructions contained in the while loop.

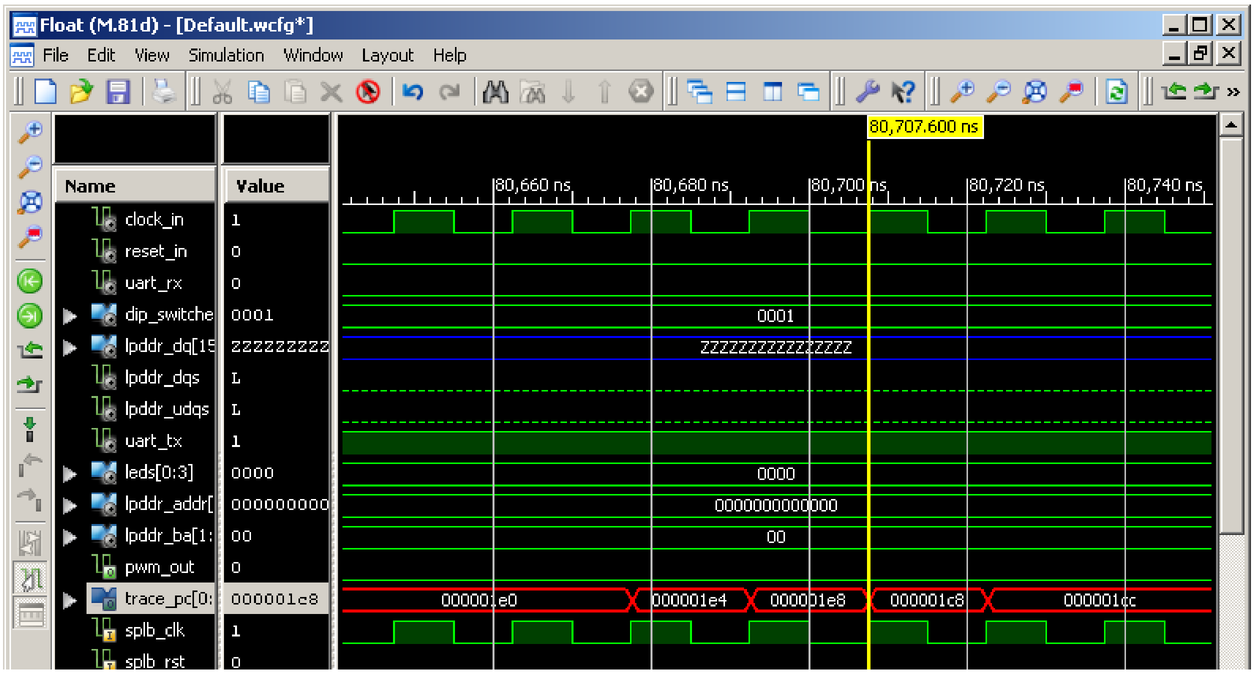


Figure 9 - Trace Program Counter viewed in ISim

19) There are many more MicroBlaze signals which can be observed.

20) Close **ISim** when finished.

21) Close **SDK**.

22) In Project Navigator, switch back to the **Implementation** View by clicking the Implementation radio button.

That concludes this tutorial. We have now simulated a MicroBlaze processor using the ISim Simulator.

IV. Getting Help and Support

Evaluation Kit home page with Documentation and Reference Designs

http://em.avnet.com/s6microboard

Avnet Spartan-6 LX9 MicroBoard forum:

http://community.em.avnet.com/t5/Spartan-6-LX9-MicroBoard/bd-p/Spartan-6LX9MicroBoard

For Xilinx technical support, you may contact your local Avnet/Silica FAE or Xilinx Online Technical Support at www.support.xilinx.com. On this site you will also find the following resources for assistance:

- Software, IP, and Documentation Updates
- Access to Technical Support Web Tools
- Searchable Answer Database with Over 4,000 Solutions
- User Forums
- Training - Select instructor-led classes and recorded e-learning options

Contact Avnet Support for any questions regarding the Spartan-6 LX9 MicroBoard reference designs, kit hardware, or if you are interested in designing any of the kit devices into your next design.

- http://www.em.avnet.com/techsupport

You can also contact your local Avnet/Silica FAE.